Certificate of Express Mail
EU 833349793 US

HOST-BASED NETWORK INTRUSION DETECTION SYSTEMS

FIELD OF THE INVENTION

The present invention relates generally to the field of communications network security and in particular to computer software for detecting intrusions and security violations in a communications network.

BACKGROUND

Communications network security generally and computer network security in particular are frequently the objects of sophisticated attacks by unauthorised intruders, including hackers. Intruders to such networks are increasingly skilled at exploiting network weaknesses to gain access and unauthorized privileges, making it difficult to detect and trace such attacks. Moreover, security threats such as viruses and worms do not need human supervision and are capable of replicating and travelling to other networked systems. Such intrusions can damage computer systems and adversely affect vital interests of entities associated with the affected network.

Existing Network Intrusion Detection Systems (NIDS) are unsuitable for deployment on every host in a network due to problems that are inherent in the architecture of such NIDS. NIDS use promiscuous mode capture and analysis, which induces significant overhead on the system and are vulnerable to insertion and evasion attacks.

Ptacek, Thomas H., and Newsham, Timothy N., "Insertion, Evasion and Denial of Service: Eluding Network Intrusion Detection", (http://secinf.net/info/ids/idspaper/idspaper.html), describe further details, including network intrusion detection.

SUMMARY

In accordance with one aspect of the invention, there is provided a method of detecting an intrusion in a communications network. The method comprises the steps of: scanning data packets processed by a transport layer of a network protocol associated with the communications network using signatures from a repository of the

signatures; determining if the scanned data packets are malicious; and taking at least one action if any data packets are determined to be malicious.

The actions may comprise at least one of interrupting transmission of any data packets determined to be malicious to the application layer of the network protocol, comprises logging of errors related to any data packets determined to be malicious, modifying firewall rules of a host computer if any data packets are determined to be malicious, informing a network administrator any data packets are determined to be malicious, intimating the transport layer terminate an existing connection related to any data packets determined to be malicious, blocking network access to a source of any data packets determined to be malicious, terminating an application of an application layer if any data packets are determined to be malicious, and notifying an application of an application layer if any data packets are determined to be malicious.

The method may further comprise the step of transmitting to the application layer any data packets determined not to be malicious.

The method may further comprise the step of processing data packets from the transport layer.

The method may further comprise the step of determining if the protocol is monitored.

The scanning and determining steps may be implemented using a scan module.

At least one application receive queue (ARQ) may function intermediate the transport layer and the application layer. The scanning step may be carried out between the transport layer and the at least one application receive queue (ARQ).

The method may further comprise the step of obtaining data from at least one application receive queue (ARQ). The at least one application receive queue may function directly intermediate the transport layer and the application layer. The scanning step may be performed on data packets in the at least one application receive queue (ARQ).

The method may further comprise the step of dispatching the data packets to one or more handlers for scanning, if the protocol is monitored.

The scanning and determining steps may be implemented using a scan daemon.

The method may further comprise the step of generating fake responses.

In accordance with another aspect of the invention, there is provided a method of preventing an intrusion in a communications network. The method comprises the steps of: disabling a network interface of a host if an idle time expires; determining if any packets are to be transmitted; and enabling the network interface if at least one packet is determined to be available to be transmitted.

In accordance with further aspects of the invention, a system for detecting an intrusion in a communications network and a computer-readable medium containing programmed instructions arranged to detect an intrusion in a communications network are disclosed, implementing the above method of detecting.

In accordance with still further aspects of the invention, a system for preventing an intrusion in a communications network and a computer-readable medium containing programmed instructions arranged to prevent an intrusion in a communications network are disclosed, implementing the above method of preventing.

BRIEF DESCRIPTION OF THE DRAWINGS

A small number of embodiments of the invention are described hereinafter with reference to the drawings, in which:

Fig. 1 is a functional block diagram of Host-Based Network Intrusion Detection Systems (HNIDS);

Fig. 2 is a functional block diagram of a Scan Module (SM) of Fig. 1;

Fig. 3 is a functional block diagram of a Scan Daemon (SD) of Fig. 1;

Fig. 4 is a diagram illustrating the normal flow of packets up a protocol stack and the processing that is done at each layer;

Fig. 5 is a diagram illustrating the normal flow of packets up a protocol stack in which a HNIDS is located between the Transport Layer and the Application Layer;

Fig. 6 is a flow diagram illustrating a process for the embodiment shown in Fig. 5;

Fig. 7 is a diagram similar to that of Fig. 5 in which a HNIDS monitors the Application Receive Queue (ARQ);

Fig. 8 is a flow diagram illustrating a process for the embodiment shown in Fig. 7;

Fig. 9 is a flow diagram illustrating a process for the Idle-Time Processing Module (ITPM) of Fig. 1; and

Fig. 10 is a functional block diagram of a HNIDS with provisions for fake services.

DETAILED DESCRIPTION

Methods, systems, and computer program products are disclosed for detecting an intrusion in a communications network. Also, methods, systems, and computer program products are disclosed for preventing an intrusion in a communications network. In the following description, numerous specific details, including network configurations, network protocols, programming languages, and the like are set forth. However, from this disclosure, it will be apparent to those skilled in the art that modifications and/or substitutions may be made without departing from the scope and spirit of the invention. In other circumstances, specific details may be omitted so as not to obscure the invention.

The methods for detecting an intrusion in a communications network may be implemented in modules. Likewise, the methods for preventing an intrusion in a communications network may be implemented in software. A module, and in particular its functionality, can be implemented in either hardware or software. In the software sense, a module is a process, program, or portion thereof, that usually performs a particular function or related functions. Such software may be implemented in Java, C, C++, Fortran, for example, but may be implemented in any of a number of other programming languages/systems, or combinations thereof. In

the hardware sense, a module is a functional hardware unit designed for use with other components or modules. For example, a module may be implemented using discrete electronic components, or it can form a portion of an entire electronic circuit such as an Field Programmable Gate Arrays (FPGA), Application Specific Integrated Circuit (ASIC), and the like. A physical implementation may also comprise configuration data for a FPGA, or a layout for an ASIC, for example. Still further, the description of a physical implementation may be in EDIF netlisting language, structural VHDL, structural Verilog or the like. Numerous other possibilities exist. Those skilled in the art will appreciate that the system can also be implemented as a combination of hardware and software modules.

Deploying a Network Intrusion Detection System (NIDS) on every host in a network substantially increases the security of the entire network. The embodiments of the invention disclose architectures that differ from existing NIDS architecture in that HNIDS architecture does not work on passive protocol analysis using promiscuous mode capture, thereby facilitating the use of NIDS on every host in the network. The embodiments presented hereinafter are not intended to be, or considered to be, a complete list of possible embodiments of the invention.

Overview

The embodiments of the invention disclose a "Host-based Network Intrusion Detection System" (HNIDS) that allows each host in a network to run network intrusion detection software, in a manner analogous to anti-virus software. The architecture enables every system on the network to act as an autonomous entity in detecting and managing intrusions.

Every system that is able to communicate over a network must use a communication protocol (e.g., the TCP/IP protocol is commonly and widely used). The Network Layer (IP in the case of the TCP/IP protocol) handles fragmentation, and the Transport Layer (TCP and UDP in the case of the TCP/IP protocol) of the communication protocol take care of reordering and reassembling packets, as necessary. Once this processing is complete, data is submitted to the application layer.

HNIDS leverages the behaviour of the protocol stack and scans data for malicious content, after the network and transport layers have completed processing the data. Thus, HNIDS works on the data in its entirety, thereby mitigating the problems of insertion and evasion attacks as well as latency and overhead that are associated with existing NIDS. HNIDS scans only the data that is destined for the system on which HNIDS is used and does not use passive protocol analysis and promiscuous mode capture.

To further explain the specifics of this architecture, two different embodiments of HNIDS are set forth. In one embodiment, HNIDS scans the data before the data is submitted to the application by the transport layer. Logically, HNIDS sits between the transport and the application layers of the communication protocol (e.g., TCP/IP). In another embodiment, HNIDS monitors the application receive queue (ARQ) for incoming data and scans the data for malicious content, as and when the data arrives.

HNIDS comprises a feature for proactively preventing intrusions, thereby acting as an "Intrusion Prevention System". This is achieved by introducing the concept of "idle time", whereby the network interface is disabled after the expiration of the idle time. Idle time is the time duration during which no packet is transmitted from the system. Since the interface is disabled, the system does not process any packets from the network. Effectively, this is the same as unplugging the system from the network, preventing intrusion related activities during the off-hours when no one is using the system (e.g., at night). The network interface is enabled again when there is a packet to be transmitted to the network, which indicates that the user is present and performing some network related activity. The resulting system with the "idle-time" feature can also be termed as "Host-based Network Intrusion Detection and Prevention System".

Recently, the use of fake services (so as to lure the attackers into a trap) is gaining importance. HNIDS may also comprise provisions for setting up fake services.

The embodiments of the invention involve HNIDS architectures that address problems associated with promiscuous mode capture and passive protocol analysis.

The architectures provide:

1) Prevention from "insertion" and "evasion" attacks;
2) Per packet analyses, and response on detection of malicious content;
3) Facilitation for usage of deception mechanisms to determine the intent of the user; and
4) Improvements to overall security of the network.

General Concept

Fig. 1 is functional block diagram 10 of a HNIDS 100 in accordance with the embodiments of the invention. The HNIDS 100 comprises a Scan Module (SM) 101 and an Idle-Time Processing Module (ITPM) 102 operating on a host computer coupled to a network. The Scan Module (SM) may be a Scan Daemon. Details of SM and SD are described in greater detail with reference to Fig. 2 and Fig. 3. Fig. 9 shows the flow diagram for the ITPM. The modules 101 and 102 are independent with separate functionality.

The ITPM module is responsible for the idle-time feature. HNIDS can function without this. ITPM provides the intrusion prevention feature.

As the HNIDS 100 is local to a system, the HNIDS 100 does not directly interface with the outside world.

The HNIDS 100 can be a separate application that is installed on the host, or the HNIDS 100 can be part of the host's network implementation.

The ITPM 102 may comprise program code to enable and disable the network interface. When the idle-time expires, the ITPM 102 disables the network interface. The interface is enabled when a packet needs to be transmitted on to the network. Enabling and disabling of a network interface is well known in the art. The network interface (physical adapter) is not shown in the drawing. The ITPM 102 comprises

code to enable/disable the network interface. Ways of accomplishing this task are well known to those skilled in the art.

This can be accomplished by providing interfaces (like IOCTL – Input/Output Control entry points) in the network driver software to enable/disable the network interface.

An Embodiment

Fig. 2 shows in greater detail the scan module (SM) 200. The scan module 200 comprises a scanning engine 202, a signature database 201 and a log database 203.

The signature database 201 contains a list of known attack signatures. This is analogous to the virus-signature database used by anti-virus systems. The scanning engine 202 uses the signatures in the signature database 201 to detect intrusions. The signature database 201 may be a plain ASCII file containing a list of signatures, although other file formats may be practiced. The signatures may be taken from the arachNIDS database.

Examples of signatures are given in Table 1, where '|' is used to enclose binary data in byte code format:

TABLE 1

"|eb 02 eb 02 eb 02|"	This event indicates that an attacker attempted to overflow one of the daemons with jmp 0x02 "stealth nops".
"GetInfo|0d|"	This event indicates that an attacker is attempting to query the NetBus remote administration tool. This legitimate administration tool is often used by attackers as a Trojan.

"|5c|IPC$|00 41 3a 00|"	This event indicates that a remote user may be attempting to open a named pipe using the IPC$ share.
"|0b 00 00 00 07 00 00 00|Connect"	This event indicates that a remote user has attempted to connect to a dagger 1.4.0 Trojan server running on Windows. This connection attempt may indicate an existing compromise.

The design of the signature database 201 is not limited per the embodiment described, but instead any suitable embodiment may be used. For example, instead of a plain ASCII file, the signatures may be stored in one or more of Microsoft Excel™ files and databases such as MySQL™ and PostgreSQL™.

The scanning engine 202 comprises program code to scan the data using the signature database 201 for the presence of a signature and take suitable action/s if any signature is found. In the embodiment of Fig. 2, relevant details may be logged to the log database 203. The action to be taken after the discovery of malicious data is not limited to the logging of errors. Other possible actions comprise modifying desktop firewall rules and informing the remote administrator. Other actions still may be practiced alone or in combination.

Fig. 3 illustrates the scan daemon (SD) 300. The Application Receive Queue (ARQ) is the queue from where the application takes its data. The scan daemon 300 comprises program code 302 to monitor the ARQ for data and subsequently analyse the data. Typically, the code 302 comprises protocol handlers 302a ..... 302n corresponding to different application protocols. A handler 302a ..... 302n is activated only for the protocol ports that are configured to be monitored. For example, if HTTP 302a and FTP 302b are configured to be monitored, then only handlers 302a, 302b for these protocols are activated. These handlers 302a, 302b use the signature database 301 to scan the data. If a match is found, appropriate errors may be logged to the log database 303.

The embodiment of Fig. 3 is not the only possible embodiment for the scan daemon 300. Other actions taken may comprise modifying firewall rules to prevent reception of packets from the offending host and intimating the transport layer to tear down the existing connection, for example.

Fig. 4 shows the flow of packets up the protocol stack 400. The TCP/IP protocol stack is shown for illustration only. Other protocol stacks that may be practiced comprise any protocol stack that follows a layered model with a clear demarcation between the Transport and Application layers.

The physical medium 410 provides packets to the link layer 412, which in turn provides the packets to the network layer processing 414. From there, the packets go to the transport layer processing 416. The transport layer 416 copies the data to the Application Receive Queue 418, which is typically a socket queue. The application layer 420 then copies the data from the ARQ 418 and uses the data.

Fig. 5 shows an embodiment of the invention. The physical medium 510, the link layer 512, the network layer 514, and the transport layer 516 correspond to the respective features 410, 412, 414, and 416 of Fig. 4. The link layer 512 may be Ethernet, Token Ring, a wireless network, and other suitable networks, Ethernet and Token Ring are named just for illustration. The link layer may be any of a number of networks, provided that the transport layer and the application layer of the network implementation are clearly demarcated, which is usually the case. The network layer 514 may be IP. The transport layer 516 may be TCP/UDP. The Host-Based Network Intrusion Detection System (HNIDS) 530 functions between the transport layer 516 and the application layer 520. The HNIDS 530 uses the scan module (SM) 200 of Fig. 2 in this embodiment. The HNIDS 530 preferably interfaces between the transport layer 516 and the ARQ 518.

Fig. 6 is a flow diagram summarizing the process 600 for the embodiment of Fig. 5. A packet is received by the HNIDS 530 from the transport layer 516 in step 601. The HNIDS 530 verifies if the network protocol is monitored in step 602. If the protocol

is not monitored (NO), HNIDS 530 passes the data to the corresponding application 520 in step 603. If the protocol is monitored (YES), then the scanning engine 202 scans the data using the signature database 201 in step 604. In decision step 605, a check is made to determine if the data is malicious. If the data is malicious (YES), the data is not passed to the application 520 (the data is not put into the ARQ), the associated connection is dropped and errors are logged in step 606. Dropping the connection means that the network connection with the remote host is torn down. The system can continue servicing requests on other existing/new connections. However, in step 606, these are not the only possible actions that can be taken, when the data is found to be malicious. Other possible actions comprise blocking access to the attacking host, blocking network access from the attacking host, and notifying the system administrator. Still other actions may be practiced. If the data is not malicious (NO), the data is passed to the corresponding application 520 in step 607.

Another Embodiment

Fig. 7 shows another embodiment. The physical medium 710 and the layers 712, 714, 716 of Fig. 7 correspond to the respective medium 410 and layers 412, 414, 416 of Fig. 4. Data is passed from the transport layer 716 to the Application Receive Queue (ARQ) 718, before being passed on to the application layer 720. The HNIDS 730 of the further embodiment monitors the Application Receive Queue (ARQ) 718, as indicated by the arrow between the HNIDS 730 and the ARQ 718. The arrow from the HNIDS 730 to the transport layer 716 indicates the HNIDS 730 may inform/instruct the transport layer 716 to tear down a connection with a remote host, if desired (or to initiate any other appropriate action). The arrow from the HNIDS 730 to the application layer 720 indicates the HNIDS 730 may inform/instruct the application 720 so as not to process the packet, to reset/free resources associated with the particular malicious connection, or even to kill the application 720, if necessary. The HNIDS 730 uses the scan daemon 300 of Fig. 3 in this embodiment.

Fig. 8 is a flow diagram summarizing the process 800 as per the above embodiment. The HNIDS 730 picks up data from the ARQ 718 and analyses the data in step 801, as and when the data arrives. In decision step 802, the HNIDS 730 determines if the protocol is monitored. If the protocol is not monitored (NO), the HNIDS 730 does

nothing in step 803. If the protocol is monitored (YES), the scanning daemon 300 dispatches the appropriate protocol handler 302a ... 302n to scan the data using the signature database 301 in step 804. In decision step 805, a check is made to determine if the data is malicious. If the data is not malicious (NO), the handler does nothing in step 806. That is, no action is required in step 806. If the data is malicious (YES), the protocol handler takes suitable action in step 807, such as tearing down the appropriate connection, killing the application, notifying the application, blocking network access from the attacking host and logging the relevant details. Any one or more of these actions may be implemented. Other actions may be implemented.

Idle-Time Processing Module

Fig. 9 is a flow diagram summarizing the process 900 for the idle-time processing module 102 of Fig. 1. When the idle time expires in step 901, the network interface is disabled in step 902. In decision step 903, a check is made to determine if there is a packet to be transmitted. If there is an indication for a packet to be transmitted (YES), the network interface is enabled again in step 904. Enabling and disabling of network interfaces is well known in the art. Otherwise, if there is no packet to be transmitted in step 903 (NO), processing returns to step 903.

The ITPM and the SCAN module are separate modules. The ITPM is responsible for the idle-time feature. HNIDS can function without this. ITPM provides an intrusion prevention feature.

The ITPM module is a separate component of the HNIDS that is responsible for providing an intrusion prevention feature in the HNIDS. The ITPM offlines the host if there are no packets transmitted from the system for a considerable period of time.

Further Embodiment

Fig. 10 is a functional block diagram of the HNIDS 1000 with the ability to fake services. The HNIDS 1000 comprises an Idle-Time Processing Module 1001, a Scan Module (SM) or Scan Daemon (SD) 1002, and a Fake Services Daemon (FSD) 1003. The modules 1001,1002 and 1003 are independent modules with their own specific

functionality. The modules 1001 and 1002 are the same as explained in relation to Fig. 1.

The FSD 1003 contains program code to fake services. The services that need to be faked are configurable. Depending upon the fake services that are configured, the FSD 1003 spawns appropriate handlers. These handlers are actually fake daemons that listen on the appropriate ports for connection requests. These daemons are not full-fledged applications, but are used for generating fake responses to fool the attacker and log the relevant details. As an example, an HTTP server may be configured as a fake service. The FSD 1003 spawns a fake HTTP daemon, which listens for connection requests on the HTTP port (80). As and when a connection request arrives on this port, relevant details such as the source IP address, hardware address, and the like are logged, and a (fake) response sent to the requesting host.

Although a small number of embodiments of the invention have been described in detail, other embodiments are possible.

Computer Implementation

The embodiments of the invention may be implemented using a computer. In particular, the processing or functionality described above and depicted in Figs. 1-10 can be implemented as software, or a computer program, executing on the computer. The method or process steps disclosed for detecting an intrusion in a communications network are effected by instructions in the software that are carried out by the computer. Likewise, the method or process steps disclosed for preventing an intrusion in a communications network may be effected by instructions in the software that are carried out by the computer. The software may be implemented as one or more modules for implementing the process steps. A module is a part of a computer program that usually performs a particular function or related functions. Also, a module can be a packaged functional hardware unit for use with other components or modules.

In particular, the software may be stored in a computer readable medium, including the storage devices described hereinafter. The software is preferably loaded into the

computer from the computer readable medium and then carried out by the computer. A computer program product includes a computer readable medium having such software or a computer program recorded on the medium that can be carried out by a computer. The use of the computer program product in the computer preferably effects an advantageous system for detecting an intrusion in a communications network in accordance with the embodiments of the invention. Likewise, a system for preventing an intrusion in a communications network may be implemented.

The computer system can be connected to one or more other computers via a communication interface using an appropriate communication channel such as a modem communications path, a computer network, or the like. The computer network may include a local area network (LAN), a wide area network (WAN), an Intranet, and/or the Internet. The computer may include a central processing unit(s) (simply referred to as a processor hereinafter), a memory which may include random access memory (RAM) and read-only memory (ROM), input/output (IO) interfaces, a video interface, and one or more storage devices. The storage device(s) may include one or more of the following: a floppy disc, a hard disc drive, a magneto-optical disc drive, CD-ROM, DVD, magnetic tape or any other of a number of non-volatile storage devices well known to those skilled in the art. The program for detecting an an intrusion in a communications network may be recorded on such a storage unit and read by the computer into memory; the same applies to a program for preventing such an intrusion. Each of the components of the computer is typically connected to one or more of the other devices via a bus that in turn can comprise data, address, and control buses. While a system using a processor has been described, it will be appreciated by those skilled in the art that other processing units capable of processing data and carrying out operations may be used instead without departing from the scope and spirit of the invention. The idle-time processing module 102 and the scan module 101 of the HNIDS 100 may be implemented using such a computer.

The described computer system is simply provided for illustrative purposes and other configurations can be employed without departing from the scope and spirit of the invention. Computers with which the embodiment can be practiced include IBM-PC/ATs or compatibles, one of the Macintosh (TM) family of PCs, Sun Sparcstation

(TM), a workstation or the like. The foregoing are merely examples of the types of computers with which the embodiments of the invention may be practiced. Typically, the processes of the embodiments, described hereinafter, are resident as software or a program recorded on a hard disk drive as the computer readable medium, and read and controlled using the processor. Intermediate storage of the program and intermediate data and any data fetched from the network may be accomplished using the semiconductor memory, possibly in concert with the hard disk drive.

In some instances, the computer program may be supplied to the user encoded on a CD-ROM or a floppy disk, or alternatively could be read by the user from the network via a modem device connected to the computer, for example. Still further, the software can also be loaded into the computer system from other computer readable medium including magnetic tape, a ROM or integrated circuit, a magneto-optical disk, a radio or infra-red transmission channel between the computer and another device, a computer readable card such as a PCMCIA card, and the Internet and Intranets including email transmissions and information recorded on websites and the like. The foregoing is merely an example of relevant computer readable mediums. Other computer readable mediums may be practiced without departing from the scope and spirit of the invention.

In the foregoing manner, methods, systems, and computer program products have been disclosed for detecting an intrusion in a communications network. Also, methods, systems, and computer program products have been disclosed for preventing an intrusion in a communications network. The detailed description provides preferred exemplary embodiments only and is not intended to limit the scope, applicability, and/or configuration of the invention. Rather, the detailed description of the preferred exemplary embodiments provides those skilled in the art with enabling descriptions for implementing the preferred exemplary embodiments of the invention. It should be understood that various changes and/or substitutions may be made in the function and arrangement of elements without departing from the scope and spirit of the invention as set forth in the appended claims.